Journal of Scheduling
https://doi.org/10.1007/s10951-023-00801-w

Real-world university course timetabling at the International Timetabling Competition 2019

Tomáš Müller[1] · **Hana Rudová**[2] · **Zuzana Müllerová**[3]

Accepted: 19 December 2023

Abstract
The paper discusses the organization of the International Timetabling Competition (ITC 2019), which intends to motivate further research on complex university course timetabling problems coming from practice. Thanks to the UniTime timetabling system, we have collected a strong set of benchmark instances with diverse characteristics for the competition. The key novelty lies in the combination of student sectioning with standard time and room assignment of particular course events. The paper analyzes the real-world course timetabling problems present in the competition. The characteristics of thirty competition instances are described together with their representative features, which are discussed institution by institution. The existing solvers are described and compared based on their competition, current, and time-limited results whenever available. As of October 2023, the competition website has about 490 registered users from 66 countries worldwide and is kept up to date with new results.

Keywords University course timetabling · Competition · Benchmarks · Real-world problems

1 Introduction

For many years, competitions have allowed significant advancements in the field of timetabling, moving the research toward more and more practical problems. The International Conference on the Practice and Theory of Automated Timetabling (PATAT)[1] supported many timetabling competitions over the years. A recent survey on educational timetabling benchmarks and competitions is available at Ceschia et al. (2023).

In 2002–2003, the First International Timetabling Competition considered simplified randomly generated course timetabling problems. The post-enrollment problem was solved, where course enrollments of students are pre-defined, and courses must be assigned in timeslots and rooms without any course overlap for students. For more details, see the paper by Kostuch (2005) presenting an improved version of the winning competition entry or related article written by one of the organizers (Lewis 2008).

The second competition in 2007 (McCollum et al., 2010) had two course timetabling tracks. One of them slightly extended the post-enrollment problem from the first competition (Lewis et al., 2007), and the other introduced curriculum-based timetabling based on real-world problems from the University of Udine in Italy (Gaspero et al., 2007). The curriculum contains a set of courses, which must be assigned into time slots with no overlaps. A more practical approach was also taken by the third examination timetabling track (McCollum et al., 2007), where particular problems were taken from existing institutions.

The third competition organized in 2011 related to educational timetabling was focused on high-school timetabling problems (Post et al., 2016). Again, there is a clear transition toward real-world problems, and we can see a complex common problem faced by thousands of educational institutions worldwide. An archive of around 50 real-world high-school timetabling data sets was created (Post et al., 2014), where

[1] http://patatconference.org/communityService.html

✉ Hana Rudová
hanka@fi.muni.cz

Tomáš Müller
muller@unitime.org

Zuzana Müllerová
mullerova@unitime.cz

[1] Purdue University, West Lafayette, IN, USA

[2] Faculty of Informatics, Masaryk University, Brno, Czech Republic

[3] UniTime, s.r.o., Zlín, Czech Republic

Published online: 12 April 2024

all the problems are defined using an XML format for benchmarks in high-school timetabling (Post et al., 2012).

This paper discusses the Fourth International Timetabling Competition (ITC 2019) (Müller et al., 2018). It introduced a variety of real-life university course timetabling problems coming from different parts of the world. A novel model of a complex course timetabling problem allows the specification of problems from various universities. In the competition, representative problems from ten universities worldwide were considered. However, they introduce a fraction of the institutions using UniTime,[2] a non-commercial software from which the instances for the competition were taken. Thirty competition and six test instances are available on the competition website[3]. The website allows for solution validation and provides a repository of existing solutions.

We will discuss the characteristics of the real-world course timetabling problems considered in the competition and demonstrate that the proposed model allows the encapsulation of very different features. We aim to provide a clear view of this competition and comprehensive information about its importance and results. To analyze particular instances, we have selected representative features to study their complexity with the help of existing upper and lower bounds (DSUM, 2023) of the solution quality. The final part of the paper concentrates on discussing and comparing the published solvers, including the description of the best computational results and the results achieved with the specified runtime. For a detailed specification of the competition problems, we refer to the paper (Müller et al., 2018), published at the PATAT 2018 conference, where the plenary talk about ITC 2019 was given.

The following section provides a summary of the competition organization. Section 3 contains an overview of the competition problem, with more details on the competition website. Section 4 introduces timetabling problems for each contributing university, and Sect. 5 discusses the different characteristics of these real-life problems. Section 6 provides a characterization of the benchmark instances, institution by institution, with the help of selected representative features and solution bounds. Section 7 describes and compares the existing solvers and their experimental results. The paper is concluded with a summary in Sect. 8.

2 Running the competition

The competition was announced at PATAT 2018, the 12th International Conference on the Practice and Theory of Automated Timetabling, where the competition framework and technical descriptions were published (Müller et al., 2018). The competition has its website, https://www.itc2019.org, maintained at Masaryk University by competition organizers and authors of this paper. The website describes the competition, the timetabling problem, the rules, the benchmark instances, and a few smaller test instances. It also includes a web service that validates the solutions for the competition instances and allows the valid solutions to be uploaded to the website. The solution validator is based on the UniTime solver. The best-uploaded solution for each competitor and instance is used to rank the competitors.

Three groups of benchmark instances have been published during the competition. While the order of the competitors was based on all the published benchmark instances, the instances released later in the competition had a much higher weight. Points were awarded to each solution based on the rank among the competitors and the type of the instance (early, middle, or late) using the Formula One ranking scheme. Two milestone submissions of the results for the early benchmark provided feedback to competitors during the long run of the competition. Since the competition problems were very complex, the competitors had one year to submit their solutions. Nowadays, the competition website is running, and solutions are updated. There are about 490 registered users from 66 countries, and we can see increases in these numbers all the time. In January 2020, at the end of the competition, we had 200 registered users, and interestingly, 60 new users appeared in the last six months. Twenty teams uploaded solutions to one or more competition benchmarks, and over 50 users successfully validated at least one solution as of October 2023.

3 Problem description

Our problems use a hierarchical course structure to model the presence of students in different parts of a course. A course may contain one or more course configurations, each with one or more classes that can be of different types and have an optional parent–child relationship. These classes are to be timetabled into rooms and (time) periods for a limited number of students. A class may occupy multiple periods, possibly spanning multiple days and weeks. It allows us to model classes with multiple meetings at the same time and room (e.g., Monday–Wednesday–Friday 9 am) and/or classes taught only during certain weeks of the semester (odd or even weeks, for instance). All benchmark instances have five-minute periods, going from midnight to midnight, seven days a week, and running for a given number of weeks (between 13 and 21). It permits a flexible time organization and supports various irregularities and other exceptions in class placements. Rooms have defined capacity and possibly unavailable periods. Each room may have specific travel times to other rooms to ensure attendance of classes at distant

[2] https://www.unitime.org

[3] https://itc2019.org

rooms. Each time and room for a class may have a penalty specifying how good or bad a given placement is.

Students are enrolled in courses and will be assigned to classes based on the defined course structure. A student must get one class of each type from a single course configuration, following the parent–child relationship when defined. For example, each student must get a lecture and a seminar, where only some lecture–seminar combinations are allowed. Sample course structure taken from UniTime interface is available in Fig. 1.

Finally, there are soft and hard distribution constraints of nineteen types defined on subsets of classes. Most constraints can be validated on pairs of classes, i.e., each pair that does not satisfy the constraint incurs a penalty. Examples are `SameDays` (all classes must be taught on the same day), `NotOverlap` (classes should not overlap), or `SameAttendees` (classes cannot overlap in time and must be placed so the attendees can travel between them). Some constraints may include parameters, for example, `WorkDay(S)` constraint penalizes placement of classes during the day longer than `S` time slots, or `MinGap(G)` penalizes the placement of classes closer than `G` time slots. Four types of constraints must be validated on the whole subset of classes. `MaxBreaks` constraint limits the number of breaks between classes daily. `MaxBlock` constraint limits the time scheduled without a break. `MaxDays` constraint requires classes not to spread over more than a certain number of days in the week. Finally, the `MaxDayLoad` constraints the number of class time slots each day.

There are four essential optimization criteria. The goal is to minimize penalties for time and room assignments of classes, penalties for unsatisfied soft distribution constraints, and the number of student conflicts. Minimizing the number of student conflicts is a fundamental part of the problem, which is crucial for university course timetabling. A student conflict exists if the student cannot attend a pair of his/her classes. The conflicts are not only between classes that overlap in time but also between classes that students cannot attend due to travel distances between assigned rooms.

Detail descriptions of particular features and XML specifications are available in Müller et al. (2018) and on the competition website.

4 Real-world problems from universities

All the competition instances in ITC 2019 represent real-world problems that have been collected from various universities around the world. These universities have been using UniTime for course timetabling in production for years. Out of the 100+ universities that are using UniTime in practice, we have tried to pick a decent sample of institutions that vary in size, country/continent, level of support they needed, and other aspects like how they were collecting student demand data. We have asked each institution for at least two data sets from at least two different semesters (e.g., one Fall and one Spring semester) that have already been timetabled using UniTime. The potential contributors were also provided with simple instructions on exporting UniTime's course timetabling solver data in an anonymized XML format.[4]

There were three institutions that we have worked closely with in the past and that have readily agreed to provide us with data sets (Masaryk, Purdue, and AGH). These are discussed in more detail below. The remaining institutions are summarized in the last subsection, as we have less information about them.

4.1 Masaryk University

Masaryk University (Czech Republic) uses UniTime for seven out of its ten faculties. We have included problems from three faculties with considerable differences (see Table 1).

The timetable for the Faculty of Informatics can be generated based on pre-enrollments of students into courses. Otherwise, it is a relatively standard mid-size problem with about 500 classes, each scheduled weekly or sometimes biweekly. Most courses only have one lecture and/or multiple seminars from which the students can choose. Classes are mostly scheduled once a week for a longer time, such as 2 h, representing a typical example of a European institution.

The Faculty of Sports Studies timetables are significantly influenced by traveling to various sports facilities across the city. It necessitates considering travel time between every two rooms to handle the attendance of classes with the same students. They also offer sports for students of other faculties of the university. These are not included in the problem as they are timetabled separately but may influence the availability of certain sports facilities.

In the Faculty of Education, we can see specific curricula patterns, typically composed of a pair of specializations, each representing one field of study such as Mathematics, Physics, English, or Music. Typical combinations, such as Mathematics–Physics or English–History, may involve many students. Some other, less popular combinations, like Music–Chemistry, or Art–Mathematics, have only a few students. This results in a very diverse set of student course demands. These pairs may result in many student conflicts because it is impossible to satisfy all the possible combinations.

There are two different types of problems for the Faculty of Education and Faculty of Sports Studies, representing (1) the present regular form of study and (2) distance learning (Müller and Rudová 2016). The Faculty of Education and the Faculty of Sports Studies construct separate timeta-

[4] https://help.unitime.org/exporting-solver-xml

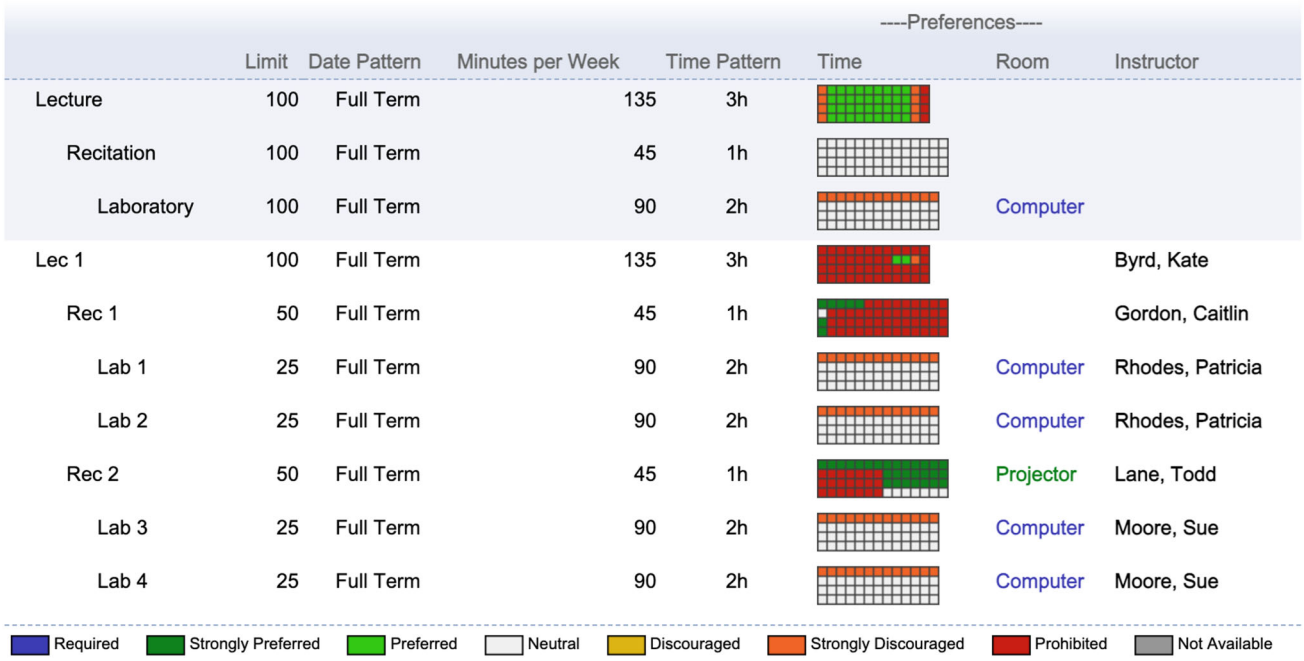

Fig. 1 Sample course structure with the lecture, recitation, and laboratory in parent–child relationship

Table 1 Competition instances from Masaryk University

Prefix	Faculty of	Terms	Specific characteristics
muni-fi	Informatics	Three terms spr16, spr17, fal17	Student pre-enrollment
muni-fsps	Sports Studies	Regular muni-fsps-spr17 Distance muni-fsps-spr17c Both muni-fspsx-fal17	Curricula Travel times between facilities
muni-pdf	Education	Regular muni-pdf-spr16 Distance muni-pdf-spr16c Both muni-pdfx-fal17	Curricula combining two fields Distance learning on Fridays and Saturdays

bles for distance learning. Here, the lifelong and combined study forms are included so students can combine studies with work duties. This problem is uncommon and complex. The students only come to school once a week or once every two weeks (e.g., every Friday or Saturday) and have all their classes during that day. Each distance learning course may have only two or three meetings during the semester, which occur on different weeks. Each of these meetings is usually modeled as a single class and linked with the other meetings using distribution constraints (e.g., required `DifferentWeeks`). It results in a very irregular pattern of timetables being different each Friday or Saturday during the semester. Consequently, these problems are often very hard, which is also reflected by the high gap between the best-known solution and its lower bound (DSUM, 2023) (47.53 % for muni-fsps-spr17c, 50.38 % for muni-pdf-spr16c, and 64.34 % for muni-pdfx-fal17).

4.2 Purdue University

Purdue University uses automated timetabling for all its departments together (Rudová et al., 2011), which means that we can see a large-scale problem representing all courses of a large public university with about 40,000 students. The timetable construction starts with timetabling for the large lecture rooms and large active learning classrooms (LLR), which the university shares and are timetabled centrally. Active learning rooms have specific features to facilitate active learning, like variable seating arrangements, dual projections, or additional equipment. In the LLR problem, there are only a few courses for each student, but the room utilization is very high since large rooms for hundreds of students represent a scarce and expensive resource. Individual departmental problems are solved later, on top of this large lecture

room problem. As the last one, the problem with shared computer laboratories is solved. In this problem, there are a lot of symmetries as many computer laboratories share the same characteristics, so it is easier to find a good solution.

At Purdue, we can also see a typical example of an American university where classes are taught several times a week at the same time and room, for instance, Monday, Wednesday, and Friday at half-hour (7:30 am, 8:30 am, ... 4:30 pm). Also, courses may be taught using different patterns, e.g., two times a week for ninety minutes or three times a week for one hour.

At Purdue, courses have a rich structure. For example, an introductory Biology course is offered to most first-year students at the university. Such a course may have a couple of large lectures needed to cover the student demand. Besides a lecture, each student may need to take a laboratory and a seminar, which are typically taught for much smaller groups of students. The parent–child relations may be used to link students of particular laboratory and seminar pairs together so that the same instructor can teach them. The course can also be offered in multiple configurations.

In contrast with many other problems discussed in this chapter, there are neither curricula nor pre-enrollments. The timetable is constructed based on last-like semester's course enrollments (e.g., timetable construction for Fall 2019 used real course enrollments for Fall 2018 to ascertain student demand and enrollment patterns).

Purdue University buildings are located on a large campus, so timetabling needs to consider non-negligible travel times, which are estimated based on the GPS coordinates of each room.

The competition instances contain just the large lecture and large active learning rooms problem (LLR, instance pu-llr-spr17), five departmental problems built on top of the LLR problem (instance pu-d5-spr17, with relevant LLR classes fixed in time and space), nine departmental problems including LLR (instance pu-d9-fal19), and the whole problem including all departments (pu-proj-fal19) that was used for projection simulation for Fall 2019.

4.3 AGH University of Science and Technology

AGH University of Science and Technology from Poland builds course timetables separately for each faculty. Still, they share some resources, and some faculties provide many courses for students outside their faculty. For instance, in our benchmarks, the Faculty of Humanities has almost two-thirds of the classes for students of other faculties. The data are structured so that the courses for students from these faculties can be managed and timetabled separately.

AGH uses relatively rigid curricula, only containing mandatory and elective courses. In the original (UniTime) problem, students of the same curriculum are kept together and attend the same classes. There are no student conflicts allowed to be created by the solver. Curriculum reservations are used to restrict students of certain curricula to particular classes. In recent years, AGH has been moving away from this model, making use of pre-enrollments and student scheduling capabilities of UniTime. They still use curricula for students with no pre-enrollment, and curriculum or student group reservations allow them to direct students to specific classes in necessary cases.

The competition data contain four Spring 2017 instances, each for an individual faculty (combining courses for both internal and external students). The Faculty of Mining Surveying and Environmental Engineering (agh-ggis) and the Faculty of Geology, Geophysics, and Environmental Protection (agh-ggos) also have a mix of regular and part-time students. The Faculty of Physics and Applied Informatics (agh-fis) and the Faculty of Humanities (agh-h) have many courses for students of other faculties (offering 46% and 73% of classes for outside students, respectively). The Fall 2017 instance (agh-fal17) contains six faculties (the four faculties from Spring plus the Faculty of Drilling, Oil, and Gas, and the Faculty of Energy and Fuels), all loaded together, offering 78 student programs and allowing for some cooperation between faculties.

4.4 Other Universities

The Maryville University in the USA (prefix mary) presents problems of another American university, but it is much smaller than Purdue and introduces a more typical user of UniTime. All courses are timetabled simultaneously, with a similar organization of times and courses.

The Universidad Yachay Tech, Ecuador, represents one of the smallest problems in the competition, but it has the highest room utilization in our set of instances. It is based on curricula, with most classes meeting weekly for 90, 120, or 180 min during the semester. There are only 28 rooms shared between 5 departments.

Four institutions from Asia are in the competition, three of which have no students in the data. These universities decided to model student course requirements using the `SameAttendees` or `NotOverlap` distribution constraints. These are the Turkish-German University (prefix tg) and the İstanbul Kültür University (prefix iku) from Turkey, and the Lahore University of Management Sciences (prefix lums) from Pakistan.

The Turkish-German University also has a few courses, but each course contains many classes that meet only once during the semester. It is a consequence of many teachers coming from Germany, so each class meeting is timetabled individually based on the instructor's availability.

The Lahore University of Management Sciences allows most classes to meet once or twice a week, but most classes

that meet two times a week must have one day between the meetings (Monday–Wednesday, Tuesday–Thursday, etc.). Most classes meet at the same time and room for the whole semester.

The İstanbul Kültür University provides two of the largest instances in the competition besides Masaryk, Purdue, and AGH instances. While there are no student demands and no special distribution constraints, their size makes them exceptional.

The Bethlehem University (prefix bet) from Palestine represents another mid-size university with course timetabling based on student pre-enrollments and classes meeting multiple times a week for the whole semester. While all the competition solvers provided similar results for the two instances from this university, they are among the instances with the highest gap between the best-known solution and the lower bound (DSUM, 2023).

For the University of Nairobi (prefix nbi), we have the only instance from Africa in the competition. It is a curriculum-based problem, with most classes meeting once a week and scheduled for the whole semester.

5 Benchmark instances

Early, middle, and late instances were published during the competition, introducing thirty benchmark instances. Summarized information about particular benchmark instances is available in Tables 2, 3, 4, and 5. The first column of each table specifies the *Instance* of the data set. Further characteristics are discussed in the following sections.

5.1 Size of the problem

Problem size is a prominent distinguishing characteristic. Instances considering only one school or faculty may involve about 500 classes, 2000 students, and 50 rooms. Other problems that represent timetabling for a large part of a university may consist of about 2800 classes, 35,000 students, or 200 rooms, for example. The biggest problem in the competition with almost 9000 classes covers courses of a whole university. Such problems are rarely timetabled as a single data instance in practice, except for planning, projections, and enrollment simulations.

Table 2 provides base information about the number of *Courses*, *Classes*, and *Rooms* in each instance. We can see that some of the classes are *fixed*, having only one possible time and, if they need a room, only one possible room available to them, which could be due to a strong relationship to other courses of the school that have been timetabled earlier. Such already timetabled classes, e.g., those that share an instructor, are a part of the data set with fixed placements. For example, muni-pdf-spr16c has almost half of the classes fixed. It consists of combined and distance learning courses being timetabled on top of the regular form of study (Müller and Rudová 2016). We can see the number of classes without a room (*no r.*). The number of rooms ranges from 15 to 768, resulting in significant differences in problems.

5.2 Students and course demands

The real-life data may have the student course demands collected from various sources. Some problems include pre-enrollment of students to courses. These can be very diverse and may introduce a high violation of student requests as there can be a few students found for almost any conceivable combination of two courses. Another option is to consider last year's student course enrollments (called last-like enrollments), which reflect the past year's situation, rather than providing information independent of the past timetable and courses.

On the other hand, student course requests can be based on curriculum requirements, which are typically easier to satisfy (Müller and Rudová 2016). Here, large groups of students are taking the same or very similar set of courses as they are following the same program of study. Such schools' curricula may be standard with some compulsory and optional courses. However, some instances, such as muni-pdf-spr16, may contain hundreds of curricula. Many students are expected to have two specializations, with some combinations being far less popular than others and having only a few students. Such curricula may be more challenging to satisfy.

Table 2 also provides information about students. For each instance, it shows the number of *Students*, the average number of courses a student is requesting (*St.courses*), and the average number of classes a student is to be enrolled into (*St.classes*). Instance iku-fal17 represents an example of a school where no student data are available, and student needs are specified using the `SameAttendees` or the `NotOverlap` constraints instead. In the remaining instances, there are typically a few thousand students in each of them. The highest numbers of students are in Purdue University problems. The pu-llr-spr17 instance has 27,018 students, but it also has a small average number of courses and classes per student because it only contains large classes that are timetabled centrally (Rudová et al., 2011). All Purdue problems are combined in the pu-proj-fal19 instance with the highest number of students, 38,437. The very high number of classes of a student (17.35 for muni-pdf-spr16c, 29.92 for agh-ggis-spr17) results from a small number of weeks per class, as it is seen in Table 4. In these problems, many courses are split into many individual classes that meet irregularly during the semester.

Table 2 Base information about benchmark instances

Instance	Basic information					Students		
	Courses	Classes	fixed	no r.	Rooms	Students	St.courses	St.classes
agh-fis-spr17	340	1239	543	588	80	1641	8.17	16.20
agh-ggis-spr17	272	1852	332	152	44	2116	6.98	29.92
bet-fal17	353	983	79	119	62	3018	6.24	9.08
iku-fal17	1206	2641	530	246	214	0	–	-
mary-spr17	544	882	63	85	90	3666	2.88	2.90
muni-fi-spr16	228	575	128	75	35	1543	6.24	10.06
muni-fsps-spr17	226	561	191	133	44	865	7.76	11.60
muni-pdf-spr16c	1089	2526	1132	200	70	2938	8.72	17.35
pu-llr-spr17	687	1001	318	34	75	27,018	3.02	3.40
tg-fal17	36	711	74	15	15	0	–	-
agh-ggos-spr17	406	1144	97	66	84	2254	7.01	13.94
agh-h-spr17	234	460	2	28	39	1988	2.60	4.18
lums-spr18	313	487	3	1	73	0	–	-
muni-fi-spr17	186	516	63	11	35	1469	6.22	10.30
muni-fsps-spr17c	116	650	32	14	29	395	6.98	32.94
muni-pdf-spr16	881	1515	159	129	83	3443	9.20	10.04
nbi-spr18	404	782	41	29	67	2293	6.03	12.46
pu-d5-spr17	212	1061	115	99	84	13,497	1.45	2.46
pu-proj-fal19	2839	8813	2809	953	768	38,437	4.71	6.95
yach-fal17	91	417	14	0	28	821	5.07	13.14
agh-fal17	1363	5081	341	421	327	6925	8.70	20.91
bet-spr18	357	1083	97	128	63	2921	6.52	10.46
iku-spr18	1290	2782	838	215	208	0	–	-
lums-fal17	328	502	8	3	73	0	–	-
mary-fal18	540	951	186	95	93	5051	4.16	4.17
muni-fi-fal17	188	535	60	4	36	1685	6.59	10.43
muni-fspsx-fal17	515	1623	443	369	33	1152	8.87	21.82
muni-pdfx-fal17	1635	3717	312	269	86	5651	9.84	15.94
pu-d9-fal19	1154	2798	449	108	224	35,213	3.51	4.37
tg-spr18	44	676	47	14	18	0	–	-

5.3 Distribution constraints

Instances may differ by the importance and the amount of distribution constraints. Some schools may use many distribution constraints, while others do not rely on them so heavily. For example, some problems have the `WorkDay`, the `MaxBlock`, and the `MaxDayLoad` constraints to provide good schedules for instructors, while other problems may not contain these three particular constraints at all.

Table 3 provides information about the distribution constraints. The total number of hard and soft distribution constraints is in the columns *Hard distr./Soft distr.*. The columns *Hard cl.pairs/Soft cl.pairs* show the number of pairs of classes with a (hard or soft) distribution constraint that can be binarized. We can see that distribution constraints are uncommon in some problems (pu-llr-spr17 given its size), while others rely heavily on them (tg-fal17 with constraints used to handle students). The numbers of special constraints that cannot be binarized (`MaxDays`, `MaxDayLoad`, `MaxBreaks`, and `MaxBlock`) are summarized in the column *Spec. const.*. Here, hard and soft constraints are specified using "/". It is clear that these special constraints are used only at some institutions, and their numbers are not very high. However, they are still essential to express institutions' specific needs, resulting in the acceptance of the generated timetables.

In practice, many distribution constraints are usually set on an instructor among the classes the instructor is expected to teach. Besides the `SameAttendees` that is used to ensure that the instructor can teach all his/her classes in the compe-

Table 3 Statistics about distribution constraints and domains

	Distribution constraints					Domains		
Instance	Hard distr.	Soft distr.	Hard cl.pairs	Soft cl.pairs	Spec. const.	Avg.times of a class	Avg.rooms of a class	Avg. availab. (%)
agh-fis-spr17	820	400	3521	769	10/20	117.73	15.92	34.1
agh-ggis-spr17	2202	488	29,475	2046	0/5	25.20	7.28	81.0
bet-fal17	861	390	5773	6564	49/0	23.77	25.43	84.3
iku-fal17	2237	665	26,768	1117	0/0	35.36	30.76	100.0
mary-spr17	3151	796	40,242	14,903	0/0	13.98	13.57	99.2
muni-fi-spr16	645	95	2113	1038	5/2	16.62	4.82	91.3
muni-fsps-spr17	331	69	2284	851	0/0	20.24	3.15	92.0
muni-pdf-spr16c	1456	570	12,574	3214	2/48	59.61	11.82	98.8
pu-llr-spr17	416	218	891	622	0/2	9.28	15.23	100.0
tg-fal17	459	42	79,299	17,090	0/0	25.86	4.41	88.4
agh-ggos-spr17	1181	507	6745	828	4/2	93.58	10.82	51.6
agh-h-spr17	288	111	2366	1247	25/26	236.35	25.47	42.5
lums-spr18	449	69	2942	651	0/0	43.86	27.19	66.1
muni-fi-spr17	639	60	1614	957	14/1	18.92	5.25	90.3
muni-fsps-spr17c	562	147	6952	397	0/0	124.74	5.06	87.5
muni-pdf-spr16	579	433	6836	2723	7/60	32.76	17.47	99.4
nbi-spr18	585	11	2131	29	0/0	38.09	4.83	99.4
pu-d5-spr17	1262	273	2734	6268	15/0	11.79	8.77	90.8
pu-proj-fal19	6399	1398	18,171	32,031	15/4	13.43	9.83	99.0
yach-fal17	529	116	1977	545	45/0	43.98	4.61	98.7
agh-fal17	4836	2318	45,193	13,596	62/116	75.55	10.52	57.9
bet-spr18	1004	414	7652	8111	63/0	23.17	25.15	86.2
iku-spr18	2833	655	32,505	779	0/0	32.72	27.72	100.0
lums-fal17	521	76	3938	670	0/0	43.50	26.54	63.2
mary-fal18	349	164	922	1418	0/0	11.37	15.11	99.0
muni-fi-fal17	635	152	1805	1019	18/3	16.30	4.94	90.1
muni-fspsx-fal17	1070	289	17,542	2186	0/0	67.85	4.42	77.4
muni-pdfx-fal17	2433	1068	42,650	7446	17/55	66.74	18.48	91.9
pu-d9-fal19	2039	707	4368	5966	17/2	13.89	14.24	98.7
tg-spr18	376	50	72,444	1986	0/2	23.37	5.67	87.2

tition problem, there are sometimes additional requirements provided by the instructor, like the `SameRoom`, `SameDays`, `MaxDays`, `MaxDayLoad`, or `MaxBlock`. Institutions with classes meeting multiple times a week following a standard set of time patterns (such as pu-llr-spr17) usually have far fewer distribution constraints, as the instructor load is already spread more evenly over the week by having each class meeting multiple times. Also, having a large number of classes that do not meet for the whole semester leads to additional distribution constraints as, for instance, a course that only meets three times a semester usually requires all three meetings to be on different weeks, preferring to start at the same time and be placed in the same room (muni-pdf-spr16c).

5.4 Domains of variables

The second part of Table 3 specifies statistics related to the domains of decision variables. We can see the average available class times (*Avg.times of a class*). While some problems are very limited in the time placement (specifically the USA problems represented by instances pu-llr-spr17 with 9.28 and mary-fal18 with 11.37 available times), some others have many options (agh-h-spr17 with 236.35 or muni-fsps-spr17c with 124.74). The same information is provided for rooms in *Avg.rooms of a class*. There is, again, interesting diversity,

e.g., muni-fsps-spr17 has 3.15 available rooms on average while having 44 rooms in total. It shows specific characteristics of rooms in the Faculty of Sports Studies (Müller and Rudová 2016) representing various sports facilities. The diversity is even more evident, considering the average domain size representing the combined time and room placement. Some problems have more than a thousand values available (agh-fis-spr17), while others have only tens values (muni-fsps-spr17). The average availability (*Avg. availab.*) is the percentage of class placements (time × room) that are available. A lower percentage indicates that some rooms are unavailable for timetabling at otherwise suitable times.

Most lectures at Purdue (e.g., see pu-llr-spr17 containing lectures mostly) follow either the 3 × 50 or the 2 × 75 time pattern, meaning that a class meets three times a week on Monday, Wednesday, and Friday for 50 min, or two times a week on Tuesday and Thursday for 75 min, respectively. Moreover, there are only ten available times for the 3 × 50 and seven for the 2 × 75 time pattern, which do not overlap. It leads to small domain sizes and resultant schedules that do not have gaps that cannot be used for additional classes without any need for additional constraints. On the other hand, at Masaryk, a regular lecture is two hours, and it can meet any day of the week with six to twelve possible start times depending on the faculty (six in muni-fi-fal17, twelve in muni-pdf-spr16). Furthermore, the distance learning classes can meet only once a semester in any available week (muni-fsps-spr17c). It makes for far larger domains in the problem. Again, this is another factor why these instances are more challenging to solve, having the high gap between the best-known solution and the lower bound (see the end of Sect. 4.1 for additional discussion).

5.5 Times

Table 4 provides information about patterns and utilization. The number of *Weeks* mostly ranges from 13 to 21. However, the number of minutes per meeting (*Minutes per mtg.*) can differ significantly, ranging from 63.52 to 140.5. The *Days per class* is higher for the USA schools, averaging 1.90 for pu-llr-spr17 (Rudová et al., 2011) and 1.56 for mary-fal18. The *Weeks per class* is one of the important parameters. We can see benchmark instances where this number is rather close to the number of weeks, but it can also be minimal for classes held only a few times per semester (muni-fsps-spr17c with 1.0, tg-fal17 with 1.16, agh-ggis-spr17 with 4.18, or muni-pdf-spr16c with 5.90).

5.6 Room utilization

For some benchmark instances, it is not the size of the problem as such but the high room utilization that makes it difficult to solve. In some of these instances, it is not the overall utilization, but clusters of rooms are in high demand. Large classrooms are a typical example of rooms with high utilization. In other instances, utilization may not be the critical component, and the optimization criteria are more emphasized.

The last three columns in Table 4 represent the utilization. The *Minutes per class* is the average number of minutes a class is taught during the semester (number of weeks × number of days of a week × minutes per meeting). The number of weeks is included because many classes may be taught for a few weeks only (see the extreme at muni-fsps-spr17c with 110.9, tg-fal17 with 159.8, or agh-ggis-spr17 with 450.2; the other extreme is mary-fal18 with 2809.7 or lums-fal17 with 2572.8). The *Minutes per room* is the average number of minutes a room is occupied during the semester (minutes of all classes in a room). The differences may be again relatively high. Instances muni-fsps-spr17 and muni-fsps-spr17c have 6008.9 and 2287.8 min, respectively (specific sports facilities are not so much used by the school), while bet-fal17, iku-spr18, mary-fal18, and pu-llr-spr17 occupy rooms heavily with about 20,000 min. The average goes up to 30,000 min in yach-fal17. Also, there may be very high differences in occupancy of different rooms, e.g., muni-fi-spr16 has very high utilization of standard classrooms. At the same time, specialized laboratories are almost empty because they are not primarily used for teaching. Another example is muni-pdf-spr16c with distance learning courses (Müller and Rudová 2016) where classes are scheduled on Saturdays and Fridays, and rooms are almost empty on other days.[5] The muni-pdfx-fal17 gives a much better picture as both regular and distance learning courses are included in the problem.

The *Minutes per student* is the average number of minutes a student is in a class during the semester (minutes of all classes a student attends). Some instances do not have actual students (iku-fal17, tg-fal17), while others may include only some student classes such as pu-llr-spr17 (Rudová et al., 2011). Also, some curricula may only include mandatory and elective courses (muni-fsps-spr17), leaving optional courses for the student to pick after the timetable is done. On the other hand, students may be allowed to pre-register for more courses than they will end up enrolling in (muni-fi-fal17), or a curriculum may contain additional courses to ensure no conflicts between possible options for the students (agh-fal17 or yach-fal17).

5.7 Optimization weights

Table 5 specifies the weights for a particular optimization

[5] As mentioned in Sect. 5.1, only some regular classes are included in the problem with a fixed time and room.

Table 4 Statistics about date and time patterns, and utilization

Instance	Date & time patterns				Utilization		
	Weeks	Minutes per mtg.	Days per class	Weeks per class	Minutes per class	Minutes per room	Minutes per student
agh-fis-spr17	16	104.46	1.01	10.93	1135.0	9352.1	17,441.4
agh-ggis-spr17	16	124.46	1.00	4.18	450.2	16,111.4	12,428.7
bet-fal17	16	93.38	1.38	15.81	1840.3	21,724.3	15,936.5
iku-fal17	14	123.70	1.00	14.00	1734.4	19,060.9	–
mary-spr17	16	141.88	1.51	13.83	2606.1	20,069.3	7252.5
muni-fi-spr16	15	121.45	1.00	10.89	1188.3	17,295.7	12,593.5
muni-fsps-spr17	19	90.97	1.00	9.30	731.6	6008.9	7728.2
muni-pdf-spr16c	13	140.50	1.00	5.90	527.0	16,985.6	5194.6
pu-llr-spr17	16	63.52	1.90	14.33	1659.0	21,666.7	5906.6
tg-fal17	14	132.53	1.00	1.16	159.8	7308.4	–
agh-ggos-spr17	16	95.31	1.00	10.50	1017.4	12,498.8	14,378.5
agh-h-spr17	16	113.71	1.02	7.52	781.6	8676.0	3442.4
lums-spr18	20	101.46	1.77	14.93	2566.3	17,072.1	–
muni-fi-spr17	14	124.55	1.00	12.35	1347.1	19,397.7	14,684.0
muni-fsps-spr17c	14	110.88	1.00	1.00	110.9	2287.8	3408.3
muni-pdf-spr16	13	84.93	1.00	11.60	931.2	15,911.7	8423.3
nbi-spr18	15	106.13	1.00	15.00	1592.0	17,819.8	17,695.0
pu-d5-spr17	15	86.73	1.59	14.72	1766.4	19,959.6	4312.1
pu-proj-fal19	17	87.37	1.54	14.09	1695.5	16,981.3	11,735.9
yach-fal17	16	113.21	1.22	15.03	2018.6	30,062.9	24,124.2
agh-fal17	18	106.29	1.00	8.58	879.7	11,937.9	17,722.8
bet-spr18	16	88.74	1.23	16.00	1639.3	20,585.4	16,291.3
iku-spr18	13	127.15	1.00	13.00	1652.9	19,830.6	–
lums-fal17	20	108.51	1.81	14.15	2572.8	17,562.7	–
mary-fal18	16	138.57	1.56	15.03	2809.7	22,564.4	10,978.4
muni-fi-fal17	13	122.54	1.00	11.51	1297.4	19,231.7	14,014.6
muni-fspsx-fal17	21	111.82	1.00	6.15	521.2	13,268.3	7869.8
muni-pdfx-fal17	13	131.45	1.00	6.50	575.6	22,777.9	8500.6
pu-d9-fal19	15	74.98	1.74	14.25	1733.0	19,696.0	7368.3
tg-spr18	16	137.90	1.00	1.53	213.4	7856.7	–

criteria, i.e., *Time* preferences, *Room* preferences, soft *Distribution* constraints, and the number of *Student* conflicts. The weights reflect different institutional needs of each problem and some of the differences in the data. For example, students in muni-fsps-spr17 are following their curricula with only a few choices in elective and optional courses. It is, hence, desired to avoid student conflicts between two mandatory courses or between a mandatory and an elective course. Therefore, overlapping any two classes with many students would result in a very high penalization.

In general, soft distribution preferences are weighted quite highly. However, due to a few soft distribution constraints, this is also because their average penalty is lower than the average penalties for times and rooms. The time preferences are usually more important than room preferences. Student conflict weights vary a lot as they also depend on whether the institution uses curricula, last year's student course enrollments, or pre-enrollments for their course timetabling.

Since there are some differences between the competition model and the model we have in UniTime, from which the data were collected, the optimization weights were also adjusted to ensure the competition solutions correspond with the results in UniTime. That is, the weighted average penalties for time, room, distribution, and student conflicts, respectively, should reflect the settings from the original instance in UniTime and, through that, the real needs of the institutions.

Table 5 Weights of optimization criteria

Instance Prefix	Time	Room	Distribution	Student
agh	4	1	15	5
bet	60	1	30	6
iku	1	1	30	–
lums	2	1	5	–
mary	2	1	5	10
muni-fi	3	1	10	5
muni-fsps	25	1	15	100
muni-pdf	3	1	12	20
nbi	2	1	8	4
pu-llr, pu-d5, pu-proj	3	1	16	8
pu-d9	5	1	10	20
tg	2	1	20	–
yach	5	1	10	5

6 Representative features of benchmark instances

In the previous section, we have discussed instances' characteristics feature by feature. This section will concentrate on the representative critical features discussed institution by institution. We want to explore what features make some benchmark instances harder to solve than others. To aid with this, we borrow the lower bounds provided by the winning team (DSUM, 2023) and compare them with the quality of the best existing solution (see the detailed information about results in Sect. 7). The gap between the value of the best solution and the available lower bound provides valuable information on how hard the problem is to solve. While there is only one solver capable of providing lower bounds, we believe that the provided bounds are significant, especially since the relative gap ranges from zero (proving that five instances have already been solved to optimality) to 94.46% for agh-fal17 which is the second-largest benchmark instance in the competition in both the number of classes and rooms.

6.1 Representative features

We have constructed graphs with the representative features and the gap for each benchmark instance. Also, it is essential to consider (heavy) weights of particular optimization criteria in Table 5.

The top graph in Fig. 2 shows the *Gap* (computed as $1 - \frac{\text{lowerbound}}{\text{bestcost}}$ and provided as a percentage), the number of *Unfixed classes* as a difference between the number of classes and fixed classes, the average number of classes a student is to be enrolled into (*St.classes*), and the number of pairs of classes with a hard distribution constraint that can be binarized (*Hard cl.pairs*). The middle graph shows the number of *Students*, the number of pairs of classes with a soft distribution constraint that can be binarized (*Soft cl.pairs*), the total number of special constraints `MaxDays`, `MaxDayLoad`, `MaxBreaks`, and `MaxBlock` in *Spec.const. sum*, and utilization as the average number of minutes a room is occupied during the semester (*Minutes per room*). The bottom graph contains information about the average number of available times of a class (*Avg.times of a class*), the average number of *Days per class*, the average number of *Weeks per class*, and the number of *Weeks* in each benchmark instance. In order to show the values well, the features were split into three graphs, each using a different scale.

Let us justify the selection of particular features to demonstrate in Sect. 6.2 how their values influence the problems' complexity. The *Unfixed classes* represent the number of classes whose placement needs to be found, so it is more representative than the number of all classes. The numbers *Students* and *St.classes* represent the dimension of problems regarding the students and their duties. *Hard cl.pairs* and *Soft cl.pairs* provide valuable information about the distribution constraints, which can be binarized. *Spec.const. sum* takes a look at the remaining distribution constraints. These three features allow us to see problem characteristics regarding distribution constraint requirements. *Hard cl.pairs* together with the *Minutes per room* characterize constraint satisfaction component of the problem. Similarly, *Avg.times of a class* are related to the number of options available for each class's time placement. Here, the time placement was selected (rather than the room placement or the corresponding domain size) since the time assignment is more critical than the room assignment. It is also clear from Table 5 where the time placement almost always has a higher weight. The difference between *Weeks* and *Weeks per class* shows the schedule regularity of the constructed timetable. If both values are close, the same timetable is constructed each week. Significant differences demonstrate that a different timetable is constructed each week, increasing the problem size and complexity. *Days per class* may demonstrate that a class is taught several times a week, which introduces specific patterns in the constructed timetables (e.g., the same room and time meetings each Monday–Wednesday–Friday). It may simplify the construction of the timetables without gaps and wasted spaces unavailable for other classes. Finally, high differences among weights of optimization criteria in Table 5 may strongly emphasize student sectioning when student conflicts are heavily weighted or course timetabling when the time weight is strongly supported.

6.2 Benchmark instances by institutions

We will discuss the above-proposed representative features in the context of particular benchmark instances and provide

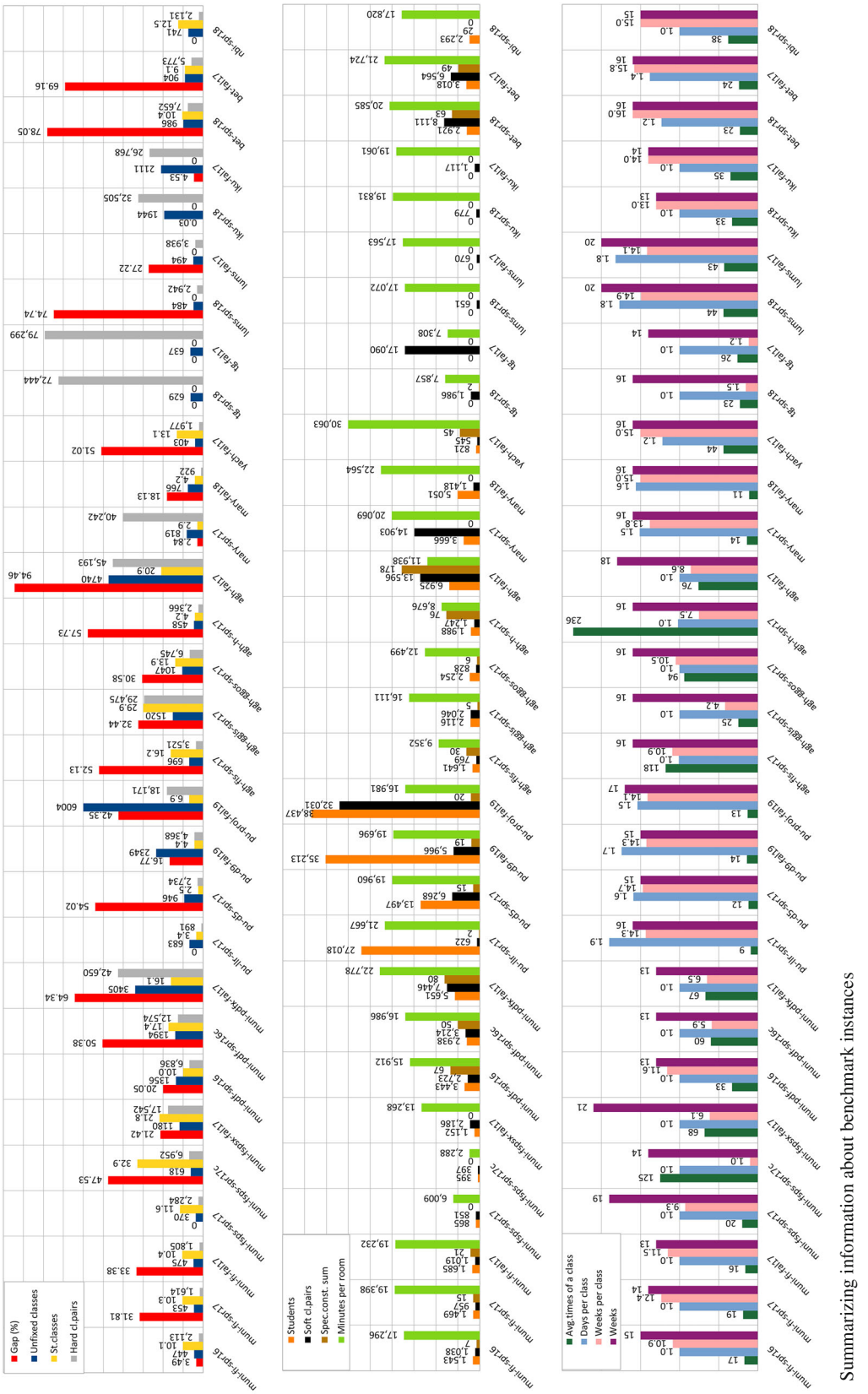

Fig. 2 Summarizing information about benchmark instances

information collected for institutions using the same structure as Sect. 4.

6.2.1 Masaryk University

The first three instances in Fig. 2 demonstrate how changes in the same problem may lead from an instance that is almost optimally solved (muni-fi-spr16) to instances where the gap is still relatively high (muni-fi-spr17 and also muni-fi-fal17). In muni-fi-spr16, we can see less than half of the special constraints, a smaller utilization, and fewer unfixed classes, resulting in a ten times smaller gap.

The next three instances from Masaryk University represent the Faculty of Sports Studies. The instance muni-fsps-spr17 is now optimally solved. There are fewer unfixed classes and students, very few hard and soft class pairs, low utilization, and no special constraints. Even though we have some discrepancies between the number of weeks and weeks per class, it does not outweigh the other characteristics. On the other hand, muni-fsps-spr17c has a very high gap representing the problem of the distance learning study. Each class is only once a week, resulting in a different timetable each week. Even though there are few students, they have many classes, and their conflicts are heavily weighted. In addition, there are many possible time placements for each class, resulting in many possible options for optimization. The instance muni-fspsx-fal17 combines features of the two earlier instances, covering both study forms. By their combination, the gap is significantly smaller than for muni-fsps-spr17c (21.42 % vs. 47.53 %), corresponding to a combination of an easier problem with a harder one. It indicates that the two problems only interact with each other a little.

For the Faculty of Education, there are three benchmark instances as well. The problem of the regular form of study muni-pdf-spr16 belongs to the ones with a smaller gap (20.05 %), corresponding to the values of the representative features, which do not show any critical issues. The gap is relatively high for the lifelong study instance muni-pdf-spr16c (50.38 %). A high discrepancy between the number of weeks and weeks per class corresponds to the need for lifelong study. The utilization is relatively high, and the average number of times of a class also allows many options for optimization. For the combined problem muni-pdfx-fal17, the gap belongs to the highest ones (64.34 %). The problem has 3,405 unfixed classes and many hard class pairs. The utilization is very high; there are many soft class pairs and special constraints, and the domains are also large (see the average possible class times). Finally, there is still a high discrepancy between the number of weeks and the average number of weeks per class.

6.2.2 Purdue University

The first instance, pu-llr-spr17, representing the large lecture room problem, is optimally solved. There are 683 unfixed classes. We have many students (27,018), but there are only 3.41 classes per student. Also, there are classes with regular patterns with 1.9 days per class.

Having pu-llr-spr17 solved to optimality was a bit of a surprise. The LLR problem at Purdue was the first problem UniTime was used for, and it was used in the original research more than two decades ago (Rudová and Murray, 2003). On the other hand, it is clear from practice that fewer available times make for easier timetabling and better timetables. More options do not make the problem easier to solve.

On the other hand, the instance pu-llr-spr17 is one of those problems that are harder to fit all the classes in, given the very high utilization of the largest rooms, and weaker on the optimization criteria, as it has a low number of classes per student and soft distribution pairs. As further discussed in Sect. 7, the instance is more suited for exact methods, such as the MIP solvers than the others.

On the contrary, the instance pu-d5-spr17 has a very high gap of 54.02 %. It looks for the placement of classes from five departments, so there are many students. The number of soft class pairs is much higher, there are more special constraints, and the regularity of the patterns is smaller.

The instance pu-d9-fal19 has a small gap of 16.77 %, while the number of unfixed classes and students is much higher. There are many soft class pairs and some special constraints. It is becoming more regular, with the average class times equal to 1.74.

Comparing the two instances, note that pu-d5-spr17 has five departmental problems to be timetabled on top of an LLR problem, whereas pu-d9-fal19 has eight departmental problems and LLR solved together. Also, pu-d5-spr17 has different weights than the other Purdue problems, see Table 5.

The last instance, pu-proj-fal19, with all Purdue University classes, has the largest number of classes, students, and a very high number of soft class pairs, resulting in a rather large gap.

6.2.3 AGH University of Science and Technology

The AGH University of Science and Technology from Poland has quite hard instances. They all have a high discrepancy between the number of weeks and weeks per class, resulting in different timetables for different weeks in the semester. The harder ones have additional special constraints. The instance agh-fal17 with the highest gap is the big problem with many classes, including many classes for each student. There are many hard class pairs, the largest number of soft class pairs, and special constraints.

6.2.4 Other universities

Other universities are represented mainly by two instances each. The benchmarks from Maryville University (mary) are easier to solve. It has a smaller number of classes, and there is a significant pattern of regularity (the average days per class are around 1.5). For the spring instance mary-spr17, there are many hard and soft class pairs but relatively few options for the time placement.

The instance yach-fal17 from Universidad Yachay Tech represents one of the smallest benchmarks. Still, the gap is very high. We can see the very high utilization and the high number of special constraints, especially concerning the size of the problem.

Both Turkish-German University (tg) instances are now optimally solved. We see mid-size problems with no explicit students and many hard constraint pairs representing them. There are (almost) no special constraints, and the utilization is low. Timetables are different each week since there are one or two weeks per class on average, but it is not an issue when combined with a small number of options for the time placement (many classes are restricted to a specific week).

One Lahore University of Management Sciences (lum) instance has a very high gap. However, it is essential to realize that the best solution has a cost of only 95, so it is not so significant in the absolute value. Both lum instances represent smaller problems with no explicit students and a few hard class pairs. No special constraints are present, and timetables have regular patterns with around 1.8 days per class.

Two İstanbul Kültür University (iku) instances have many unfixed classes. Again, there are students represented using hard class pairs only, but now there are significantly fewer than in the two earlier problems. There are no special constraints, few soft class pairs, and timetables are the same each week. The gap is minimal, even though the instances belong to the largest ones.

The Bethlehem University (bet) instances have a very high gap. We can see a high number of soft class pairs and special constraints, especially considering that they are not large instances. In addition, the utilization is relatively high, also. Finally, we need to consider higher weights of time preferences and distribution constraints, resulting in higher costs in the order of magnitude.

The last nbi-spr18 instance, from the University of Nairobi, has a medium size. There are 741 unfixed classes and 2293 students. We cannot see any specific demands in the considered features, and the problem is solved to optimality.

7 Results of competition

Solvers of various types have approached the competition instances. The first two winning results were computed by matheuristic solvers (Mikkelsen and Holm, 2022; Rappos et al., 2022), generally applying the fix-and-optimize approach (Lindahl et al., 2018). The hybrid constraint-based metaheuristic solver UniTime ITC 2019 can now compute the second-best results (Müller, 2022). Another metaheuristic solver (Sylejmani et al., 2022) applying simulated annealing (SA) (Kirkpatrick, 1984) provides the fourth current best results. The fifth best solver relies on reformulation to MaxSAT (Lemos et al., 2022). The last successful competition solver is described in the competition paper (Er-rhaimini, 2020) only. It is based on a hybrid search with features of beam search and hill climbing (Russell and Norvig, 2020).

Thanks to the open-source prize, the source codes of three solvers are now publicly available. See GitHub repositories for the SA solver,[6] the MaxSAT-based solver,[7] and UniTime ITC 2019 solver.[8]

The authors of the winning solver also provide a reduction procedure for the ITC 2019 instances (Holm et al., 2022) since there are various redundancies in the data. These reduced instances are available from their website (DSUM, 2023), where lower bounds are also provided, demonstrating that five instances are now optimally solved.

The next part of this section describes the main ideas about solvers which are already published. In Sect. 7.2, we will compare solvers and their characteristics. Finally, Sect. 7.3 provides computational results.

7.1 Existing solvers

The winning matheuristic solver was submitted by Holm et al. (2020) and further refined by Mikkelsen et al. (2022). It relies on modeling using different graph structures in conflict graphs and a reduction algorithm, which removes redundancies in the input data by reducing fixed vertices and cliques in the graph. This graph-based formulation was published in Holm et al. (2022) where authors solved all instances with the exclusion of pu-proj-fal19. For this instance, the model did not fit into 256 GB of RAM. The work (Mikkelsen and Holm, 2022) applies the fix-and-optimize matheuristic (Lindahl et al., 2018) with adaptive updates of the neighborhood size to be fixed during the search. The parallel runs of a fix-and-optimize solver are processed under different initial solutions, with different neighborhoods, and sharing current best available solutions. Initial timetables can be generated without students, and the student enrollments are processed in a separate phase to handle problems with many students, such as the instance pu-proj-fal19. As an underlying solver, Gurobi[9] was used.

[6] https://github.com/edongashi/itc-2019

[7] https://github.com/ADDALemos/MPPTimetables

[8] https://github.com/tomas-muller/cpsolver-itc2019

[9] https://www.gurobi.com

The second matheuristic solver by Rappos et al. (2022) also applies reduction strategies in their mixed integer programming model (MIP). In the pre-processing phase, variables with a fixed value and always satisfied constraints are eliminated, and similar constraints are aggregated. The initial MIP solver run concentrates on finding a feasible solution. It starts with a subset of constraints first. Consequently, other constraints are iteratively injected, and several strategies are used to fix part of the variables. The next stage relies on fixing several variables to values from previous iterations, aiming to improve the objective function while keeping feasibility. The authors have only published the competition results. They solved all instances except for agh-fal17 due to the competition deadline. For the computation, the authors altered between two solvers, IBM CPLEX[10] and Gurobi, sometimes resulting in minor improvements.

The solver available from the competition organizer Tomáš Müller (2022) uses the UniTime solver adjusted to the ITC 2019 setting. The solver relies on the constraint-based model (Rudová et al., 2011) and consists of multiple phases to find a solution. Initially, a constructive student sectioning algorithm assigns students to classes to keep students with similar course demands together. It allows the solver to compute the number of students shared between pairs of classes. Consequently, the iterative forward search algorithm with conflict-based statistics constructs a feasible solution. Hill climbing is used to find a local optimum in the next phase. Further improvement in optimization results is processed by great deluge (Dueck, 1993), using bound restarts when a better solution cannot be found. Finally, student sectioning is improved by the great deluge with moves and swaps of single students between alternative classes of a single course.

Metaheuristics are represented by the simulated annealing solver by Sylejmani et al. (2022). The pre-processing phase precomputes possible class combinations for each course, and the worst penalties are calculated for the objective normalization. An initial solution with times and rooms for all classes set to 0, and a simple assignment of students to classes is computed in the first phase and used as input for SA. In SA, the penalty for the violated hard constraints, soft constraints, and current student conflicts is maintained separately. Initially, when hard constraints are violated, subtle changes in soft and student penalties are discouraged by the objective function. SA is processed with stochastic tunneling to normalize the evaluation function and restarts when no solution is found within several iterations. The search also focuses on constraints that are hard to satisfy during several restarts. A random walk search phase is entered to minimize their penalty if such a constraint exists.

Lemos et al. published an improved version of their competition solver in Lemos et al. (2022). The pre-processing phase includes several procedures. First, the independent sub-instances are identified. Next, students with the same course enrollment plan are merged, taking class capacities into account. Also, class domains are reduced, and redundant constraints are removed. The main phase consists of the separate course timetabling and student sectioning, completed using the MaxSAT solver TT-Open-WBO-Inc (Nadel, 2019). This decomposition reduces the size of the problem, which is critical for instances with a large number of students. Symmetry breaking is introduced for `MaxBlock` and `MaxBreak` constraints, which would otherwise make it impossible to solve even small instances. The iterative calls of the MaxSAT solver allow it to handle high memory demands of the *exactly one* constraints, which is used in the proposed MaxSAT encodings (e.g., in the iku instances). The final phase processes hill climbing to improve student sectioning by swapping and moving student clusters.

7.2 Common and different features of the solvers

When considering the characteristics of the existing solvers, they often rely on various forms of pre-processing to identify and remove redundancies in the data. As mentioned, the instances reduced by pre-processing are available at DSUM (2023). However, these may not be helpful for other solvers, such as the UniTime-based solver, which has no explicit pre-processing. There have been no measurable improvements for the UniTime-based solver when used on the reduced instances from DSUM (2023).

Given the complexity of the problems, all solvers process several search phases to decompose the problem. One of the essential components is the feasibility problem, which concentrates on the satisfaction of hard constraints in the problem. The separated or interleaved handling of course timetabling (assignment of times and rooms to classes) and student sectioning (assignment of students to classes) is often used, even though it may remove optimal solutions.

The difference between matheuristic solvers consists in their constraint handling. While Mikkelsen et al. (2022) rely on complex graph structures (Holm et al., 2022), Rappos et al. (2022) solve a relaxed version of the problem, adding new restrictions each time a found solution does not satisfy all hard constraints of the problem. It allows for a smaller model as the addition of constraints resulting in millions of equations is avoided. Different memory demands of both solvers also reflect it. While the winning solver needed 756 GB of RAM, Rappos et al. (2022) with the second best competition results used 32 GB of memory only.

As another consequence, Rappos et al. (2022) identified that their approach does not perform well for problems with a large number of students, and it is hard to obtain a feasible solution when a large number of distribution constraints are combined with a large number of classes.

[10] https://www.cplex.com

Given the increasing demands of the matheuristic solvers with the increasing problem size, we can see that the UniTime-based solver, which relies on heuristic methods, performs better on the largest benchmarks (pu-proj-fal19 and agh-fal17).

The metaheuristic methods are naturally stronger in approaching the optimization than the satisfaction component. Based on the computational results (see next section), it is also reflected by the SA solver, whose performance is weaker when there is a high number of hard class pairs (or high utilization), showing a weaker performance in constraint satisfaction.

Lemos et al. (2022) stated that their solver performs worse for instances where student conflicts are strongly penalized (compare results and weights of student conflicts in Table 5, especially for mary, muni-fsps, muni-pdf). It is related to the fact that the significant improvement in the student sectioning is processed in the final phase.

7.3 Computational results

The final results of the five finalists are presented in Table 6, where the quality of the best-found solution is presented together with the achieved number of points using the Formula One ranking scheme.

Nowadays, most of the competition results are published in the papers. We provide more detailed results in Table 7. The column *LB* contains the lower bound provided by the winning solver (Mikkelsen and Holm, 2022). Note that the lower bound available from the paper was updated by the authors on their website (DSUM, 2023). We can see that five instances are now optimally solved (costs written in bold). The column *Gap* contains the gap concerning the best available result. For each publication, we provide its best available results (*Now*), the competition results (columns *Comp.*), and the results provided with the specified runtime (*Time*) to allow a comparison from the limited runtime point of view.

The computational setup, as well as the runtimes of solvers, differs a lot. Mikkelsen et al. (2022) used a farm of high-end server computers, each equipped with up to 756 GB of RAM, two CPUs (each having 16 cores), and a Gurobi MIP solver. For the last ten competition days and the runtime-limited setup, they run the parallelized setup on each instance *once* for the entire ten days, providing results after 24 h as well (24-hour results are available in the *Time* column). Their setup consisted of a solver running a full MIP model focusing on the bound using 16 threads and six fix-and-optimize processes, each using a different configuration and four threads. Two additional Gurobi MIP solvers were used to produce a pool of initial solutions, each using four threads. Rappos et al. (2022) solved instances on virtual machines with 32 GB RAM and 4 CPU cores. They provide the competition results only, and the average runtime among all instances was 83.3 h. Müller (2022) used a computer with a single CPU and 10 cores with 64 GB memory. However, each solver run was limited to a single CPU core, 4 GB of RAM, and 2 h of runtime. The *average* results from 10 independent runs are provided in column *Time*. Sylejmani et al. (2022) used 2 CPUs, each with 16 cores and 96 GB of RAM. The *average* results of 10 runs, each taking 24 h, are provided in column *Time*. Finally, Lemos et al. (2022) performed experiments on a computer with 128 GB of RAM using a single CPU core. They provide only the *best* result among 24 runs (8 different algorithmic configurations, three different encodings), each taking 100 min, which are available in column *Time*.

Mikkelsen and Holm (2022) improved their competition results for 21 instances. There are no competition results for Müller (2022) since he was not allowed to participate in the competition as its organizer. Sylejmani et al. (2022) were able to improve results for three instances only. Rappos et al. (2022) could not solve one instance (agh-fal17), and the results were not further improved. The most significant difference has been achieved by Lemos et al. (2022), who can now solve all instances compared to 18 instances for the competition. The progress is also clear based on the first (full) paper published at PATAT 2022 (Lemos et al., 2021) and the journal publication (Lemos et al., 2022), where they have concentrated on the iterative MaxSAT solver processing.

When comparing the current best results with the competition results, the competitors' order differs by the better position of the MaxSAT-based solver. It now only slightly loses against the SA solver. The UniTime-based solver became the second-best solver. Regarding computational demands, the UniTime-based solver is superior, providing results within 2 h and using 4 GB of memory per solver run. All other solvers needed much longer runtime and memory.[11]

We also provide additional graphs comparing runs with the specified runtime (see Fig. 3). The comparison must be taken carefully since we can see results from different setups. Still, it is a valuable comparison providing results for the limited computational resources. It is clear that the SA solver has significant limits in a shorter time frame; there are eight unsolved instances. We can also see that the results of the MaxSAT-based solver are sometimes relatively weak, especially for instances with a strong emphasis on student conflicts. For (Rappos et al., 2022), the results for only five instances were computed within less than 24 h, even though some lower-quality results with shorter runtimes could have probably been provided for more instances. It is good to see that the winning matheuristic solver can provide reasonable results within 24 h. Its results are still better than the UniTime-based

[11] Note that the shorter runtime results of the MaxSAT-based solver are the best among 24 runs.

Table 6 Results of the competition

Instance	Holm et al.		Rappos et al.		Gashi et al.		Er-rhaimini		Lemos et al.	
	penalty	pts	penalty	pts	penalty	pts	penalty	pts	penalty	pts
agh-fis-spr17	**3081**	10	4557	7	6799	3	5709	5	35,139	2
agh-ggis-spr17	**35,808**	10	36,616	7	77,932	3	56,755	5	194,138	2
bet-fal17	**290,086**	10	295,427	7	299,205	5	313,812	3	–	–
iku-fal17	**18,968**	10	26,840	7	50,613	3	44,482	5	–	–
mary-spr17	**14,910**	10	15,021	7	15,894	5	16,698	3	51,147	2
muni-fi-spr16	**3756**	10	3844	7	5006	5	5207	3	19,314	2
muni-fsps-spr17	**868**	10	883	7	1938	5	4135	3	211,142	2
muni-pdf-spr16c	**36,487**	10	37,487	7	58,206	5	77,573	3	–	–
pu-llr-spr17	**10,038**	10	13,385	7	16,874	5	19,231	3	68,003	2
tg-fal17	**4215**	9	**4215**	9	8044	2	7358	3	6774	5
agh-ggos-spr17	**3055**	15	6320	11	9666	6	7725	8	79,745	4
agh-h-spr17	**23,502**	15	26,159	6	25,081	11	25,745	8	55,887	4
lums-spr18	**95**	15	114	8	107	11	178	6	820	4
muni-fi-spr17	**3825**	15	4289	11	4692	8	5433	6	18,080	4
muni-fsps-spr17c	**2596**	15	3303	11	9222	8	23,520	6	618,217	4
muni-pdf-spr16	**18,151**	15	24,318	11	40,074	6	38,826	8	310,994	4
nbi-spr18	**18,014**	15	19,055	11	26,517	8	30,309	6	49,924	4
pu-d5-spr17	**15,910**	15	18,813	11	19,440	8	20,242	6	–	–
pu-proj-fal19	**148,016**	15	561,194	6	237,909	8	176,039	11	–	–
yach-fal17	**1239**	15	1844	8	1727	11	3181	6	32,198	4
agh-fal17	186,200	15	–	–	184,030	18	**153,236**	25	–	–
bet-spr18	**348,589**	25	360,057	18	360,437	15	373,039	12	–	–
iku-spr18	**25,878**	25	36,711	18	85,969	12	70,932	15	–	–
lums-fal17	**349**	25	386	18	486	15	558	12	1151	10
mary-fal18	**4423**	25	5637	18	7199	12	6944	15	44,097	10
muni-fi-fal17	**2999**	25	3794	18	4712	15	4820	12	–	–
muni-fspsx-fal17	**17,074**	25	33,001	18	44,059	15	104,625	12	–	–
muni-pdfx-fal17	**117,412**	25	151,464	18	170,061	15	191,887	12	–	–
pu-d9-fal19	**43,006**	25	134,009	12	82,757	15	70,450	18	–	–
tg-spr18	**12,704**	25	12,856	18	15,992	15	19,738	12	31,900	10
Total		489		322		273		252		79

Bold values indicate the best solution for each instance

solver, now in 21 instances (for the best result, Mikkelsen et al. are better in 24 instances), but keeping in mind that the matheuristic solver needs a longer runtime, more memory, and more CPU cores.

8 Conclusion

The International Timetabling Competition 2019 was aimed at solving common university course timetabling problems from practice. A wide set of features was considered. The key novelty lies in the combination of student sectioning, together with standard time and room assignment of events in courses. As a part of the competition, we were able to collect an interesting set of data which has enriched further research.

While the competition is long over, ITC 2019 competition problems are still very much alive. The number of people registered on the competition website increased from 200 to 490 since the competition ended in January 2020. Before the competition deadline, 6 teams submitted their solutions. Some results are now published by 20 teams on the competition website, and we believe that these numbers will increase much more given the continuing interest of the community. Also, 3 finalists, including the winning team, published better results than they had in the competition. 25 out of 30 benchmark instances wait for their optimal solutions.

Table 7 The detailed computational results: Lower Bound from DSUM (2023); Mikkelsen and Holm (2022), Gap wrt. the best result, current results (columns Now), competition results (columns Comp.), results with specified runtime (columns Time), for Mikkelsen and Holm (2022) and Sylejmani et al. (2022) within 24 hours, for Müller (2022) within 2 hours, for Lemos et al. (2022) within 100 min; Rappos et al. (2022) provides competition results only

	LB	Gap (%)	Mikkelsen et al.			Müller		Rappos et al.	Sylejmani et al.			Lemos et al.		
			Now	Comp.	Time	Now	Time	Comp.	Now	Comp.	Time	Now	Comp	Time
agh-fis-spr17	1429	52.13	2985	3081	3463	3342	4630	4557	6799	6799	7200	35,139	35,139	35,139
agh-ggis-spr17	23,164	32.44	34,285	35,808	38,026	36,305	48,964	36,616	77,932	77,932	–	161,118	194,138	161,118
bet-fal17	89,278	69.16	289,452	290,086	319,059	289,954	294,907	295,427	299,205	299,205	–	296,015	–	296,015
iku-fal17	18,109	4.53	18,968	18,968	19,498	23,096	28,178	26,840	50,613	50,613	63,899	29,929	–	29,929
mary-spr17	14,486	2.84	14,910	14,910	14,924	15,122	15,494	15,021	15,894	15,894	19,915	51,147	51,147	51,147
muni-fi-spr16	3621	3.49	3752	3756	3766	4039	4227	3844	5006	5006	6061	19,314	19,314	19,314
muni-fsps-spr17	**868**	0	868	868	868	879	1795	883	1938	1938	9028	211,142	211,142	211,142
muni-pdf-spr16c	16,255	50.38	32,762	36,487	66,812	39,344	53,157	37,487	58,206	58,206	79,787	53,803	–	56,790
pu-llr-spr17	**10,038**	0	10,038	10,038	10,055	10,772	11,872	13,385	16,874	16,874	22,399	68,003	68,003	68,003
tg-fal17	**4215**	0	4215	4215	4215	4215	5817	4215	8044	8044	–	6774	6774	6774
agh-ggos-spr17	1982	30.58	2855	3055	4652	3385	4790	6320	9328	9666	16,256	79,745	79,745	79,745
agh-h-spr17	8945	57.73	21,161	23,502	23,883	22,161	24,867	26,159	25,081	25,081	–	55,887	55,887	55,887
lums-spr18	24	74.74	95	95	95	97	109	114	107	107	119	95	820	119
muni-fi-spr17	2549	31.81	3738	3825	3845	3953	4365	4289	4692	4692	4872	18,080	18,080	18,080
muni-fsps-spr17c	1361	47.53	2594	2596	3777	2951	4925	3303	9222	9222	16,133	618,217	618,217	618,217
muni-pdf-spr16	13,719	20.05	17,159	18,151	22,533	19,768	23,805	24,318	40,074	40,074	43,210	310,994	310,994	310,994
nbi-spr18	**18,014**	0	18,014	18,014	18,014	18,627	19,859	19,055	26,517	26,517	31,088	49,924	49,924	49,924
pu-d5-spr17	6981	54.02	15,842	15,910	17,731	15,184	15,566	18,813	19,440	19,440	20,377	15,733	–	17,513
pu-proj-fal19	67,549	42.35	147,712	148,016	219,832	117,169	124,044	561,194	237,909	237,909	–	117,177	–	126,568
yach-fal17	526	51.02	1135	1239	1717	1074	2001	1844	1727	1727	1906	32,198	32,198	32,198
agh-fal17	6522	94.46	140,194	186,200	261,826	117,627	133,755	–	184,030	184,030	–	142,687	–	142,687
bet-spr18	76,489	78.05	348,524	348,589	375,677	348,533	353,374	360,057	360,437	360,437	–	353,920	–	353,920
iku-spr18	25,855	0.03	25,863	25,878	28,436	35,184	43,082	36,711	85,969	85,969	–	45,537	–	45,537
lums-fal17	254	27.22	349	349	349	366	411	386	486	486	524	813	1151	813
mary-fal18	3546	18.13	4331	4423	4546	4795	5101	5637	7199	7199	9597	44,097	44,097	44,097
muni-fi-fal17	1890	33.38	2837	2999	3199	3166	3790	3794	4712	4712	6075	4161	–	19,683
muni-fspsx-fal17	7869	21.42	10,645	17,074	36,461	10,014	15,640	33,001	41,933	44,059	56,938	101,317	–	401,155
muni-pdfx-fal17	29,333	64.34	82,258	117,412	138,916	96,312	125,201	151,464	159,203	170,061	–	151,461	–	228,560
pu-d9-fal19	32,321	16.77	38,834	43,006	47,938	44,440	47,442	134,009	82,757	82,757	158,886	47,543	–	71,903
tg-spr18	**12,704**	0	12,704	12,704	12,704	14,548	20,418	12,856	15,992	15,992	–	31,900	31,900	31,900

Bold values indicate optimal solutions

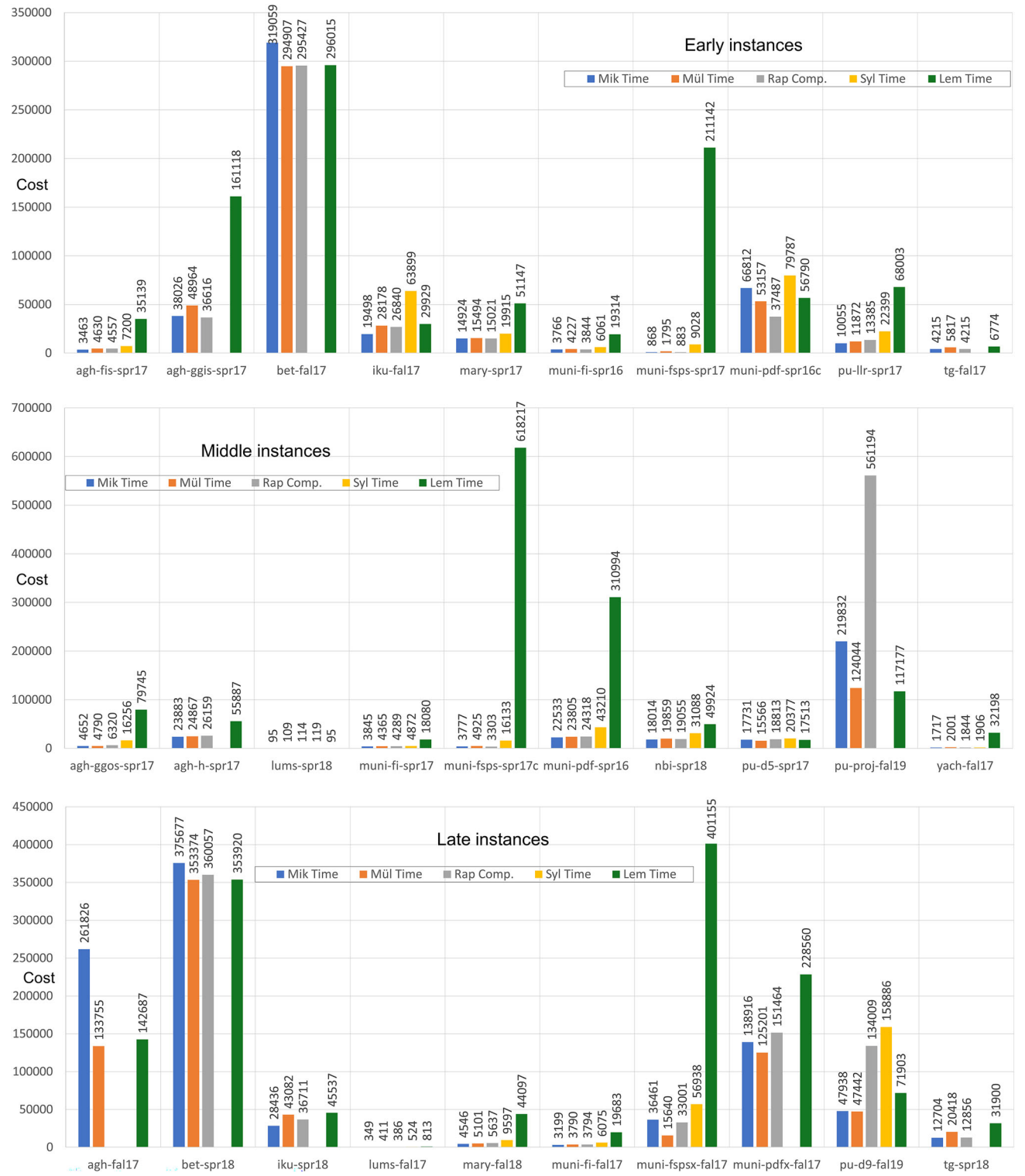

Fig. 3 Comparison of the results with the specified runtime: Mik Time (Mikkelsen and Holm, 2022) within 24 h (one run), Mül Time (Müller, 2022) within 2 h (average run), Rap Comp. (Rappos et al., 2022) with an average competition time 83.3 h, Syl Time (Sylejmani et al., 2022) within 24 h (average run), and Lem Time (Lemos et al., 2022) within 100 min (the best run)

It is great to see the development of educational timetabling studied with the help of competitions supported by the PATAT conference. In 2002, the First International Timetabling Competition considered very basic timetabling problems generated by the computer. Over the years, new competitions came with more and more realistic timetabling problems, with 2011 introducing real-life high school timetabling problems. We are proud to allow for advancements in university course timetabling, providing a diverse set of real-world problems with complex characteristics.

Acknowledgements We want to thank the sponsorship of the PATAT conference and the EURO working group on Automated Timetabling (EWG PATAT). We are pleased to thank commercial sponsors, the company ORTEC, and the Apereo Foundation for providing additional prices to the competitors. We also acknowledge Masaryk University, which provided resources for maintaining the competition website, and UniTime, s.r.o. for the itc2019.org domain.

Funding Open access publishing supported by the National Technical Library in Prague.

References

Ceschia, S., Gaspero, L. D., & Schaerf, A. (2023). Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research, 308*(1), 1–18.

DSUM data science for university management, ITC 2019. https://dsumsoftware.com/itc2019/. Accessed 23 Aug 2023

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics, 104*(1), 86–92.

Er-rhaimini, K. (2020) Forest growth optimization of solving timetabling problems. In *International timetabling competition 2019*.

Gaspero, L. D., McCollum, B., Schaerf, A. (2007). The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Tech. Rep. QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1.0, Queen's University, Belfast.

Holm, D. S., Mikkelsen, R. Ø., Sørensen, M., Stidsen, T. J. R. (2020). A MIP based approach for International Timetabling Competition 2019. In *International timetabling competition 2019*.

Holm, D. S., Mikkelsen, R. Ø., Sørensen, M., & Stidsen, T. J. R. (2022). A graph-based MIP formulation of the International Timetabling Competition 2019. *Journal of Scheduling, 25*, 405–428.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics, 34*(5–6), 975–986.

Kostuch, P. (2005). The university course timetabling problem with a three-phase approach. In E. Burke & M. Trick (Eds.), *Practice and theory of automated timetabling V* (pp. 109–125). Berlin, Heidelberg: Springer.

Lemos, A., Monteiro, P. T., Lynce, I. (2021). ITC 2019: University course timetabling with MaxSAT. In *Proceedings of the 13th international conference on the practice and theory of automated timetabling PATAT 2021*, Volume I (2020), pp. 105–128 (2021).

Lemos, A., Monteiro, P. T., & Lynce, I. (2022). Introducing UniCorT: An iterative university course timetabling tool with MaxSAT. *Journal of Scheduling, 25*, 371–390.

Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum, 30*, 167–190.

Lewis, R., Paechter, B., McCollum, B.: Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. In Cardiff working papers in accounting and finance A2007–3, Cardiff Business School, Cardiff University (2007).

Lindahl, M., Sørensen, M., & Stidsen, T. R. (2018). A fix-and-optimize matheuristic for university timetabling. *Journal of Heuristics, 24*, 645–665.

McCollum, B., McMullan, P., Burke, E.K., Parkes, A.J., Qu, R. (2007). The second international timetabling competition: Examination timetabling track. Tech. Rep. QUB/IEEE/Tech/ITC2007/Exam/v4.0/17, Queen's University, Belfast.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., & Burke, E. K. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing, 22*(1), 120–130.

Mikkelsen, R. Ø., & Holm, D. S. (2022). A parallelized matheuristic for the International Timetabling Competition 2019. *Journal of Scheduling, 25*, 429–452.

Müller, T. (2022). ITC 2019: Results using the UniTime solver. In *PATAT 2022—proceedings of the 13th international conference on the practice and theory of automated timetabling, vol. III*, pp. 243–247.

Müller, T., & Rudová, H. (2016). Real-life curriculum-based timetabling with elective courses and course sections. *Annals of Operations Research, 239*(1), 153–170.

Müller, T., Rudová, H., Müllerová, Z. (2018). University course timetabling and International Timetabling Competition 2019. In *PATAT 2018—proceedings of the 12th international conference on the practice and theory of automated timetabling*, pp. 5–31.

Nadel, A. (2019). Anytime weighted MaxSAT with improved polarity selection and bit-vector optimization. In *2019 Formal methods in computer aided design (FMCAD)*, pp. 193–202.

Post, G., Ahmadi, S., Daskalaki, S., Kingston, J. H., Kyngas, J., Nurmi, C., & Ranson, D. (2012). An XML format for benchmarks in high school timetabling. *Annals of Operations Research, 194*(1), 385–397.

Post, G., Di Gaspero, L., Kingston, J. H., McCollum, B., & Schaerf, A. (2016). The third international timetabling competition. *Annals of Operations Research, 239*, 69–75.

Post, G., Kingston, J. H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H., & Schaerf, A. (2014). XHSTT: An XML archive for high school timetabling problems in different countries. *Annals of Operations Research, 218*(1), 295–301.

Rappos, E., Thiémard, E., Robert, S., & Hêche, J. F. (2022). A mixed-integer programming approach for solving university course timetabling problems. *Journal of Scheduling, 25*, 391–404.

Rudová, H., Müller, T., & Murray, K. (2011). Complex university course timetabling. *Journal of Scheduling, 14*(2), 187–207.

Rudová, H., & Murray, K. (2003). University course timetabling with soft constraints. In E. Burke & P. De Causmaecker (Eds.), *Practice and theory of automated timetabling IV* (pp. 310–328). Berlin, Heidelberg: Springer.

Russell, S. J., Norvig, P. (2020). *Artificial intelligence: A modern approach, 4 edn*. Pearson.

Sylejmani, K., Gashi, E., Ymeri, A. (2022). Simulated annealing with penalization for university course timetabling. *Journal of Scheduling*. Published: 20 Jul 2022.